

Project-2 Report on **“OBJECT RECOGNITION AND CLASSIFICATION”**

Bachelor of Technology (B. Tech.)
in
Computer Science and Engineering



In partial fulfillment of requirements for the degree of

Submitted by

Ms. Neetu (200373)

Ms. Ritika Rathi(200381)

Under the Guidance of

Dr. Sanjeev Patwa

Department of Computer Science and Engineering

SCHOOL OF ENGINEERING AND TECHNOLOGY

Mody University of Science and Technology
Lakshmangarh, Distt. Sikar-332311

April, 2023

A C K N O W L E D G E M E N T

First of all, we would like to thank our mentor Dr. Sanjeev Patwa, for helping us starting from the beginning to the end of the development of our project. We have extended our supreme gratitude to Mody University of Science and Technology for providing such kind of opportunity for students to broaden their perception on how the real world in the field of computer science and engineering looks like as well its effort to make sure that the whole project achieve its desired goals.

We would also like to express our special thanks to Dr. Sanjeev Patwa, giving us a chance to spend our practice in his supervision and helping us in our day-to-day activities during the development time. We would like all the people who worked with us during this period with their patience and openness they created an enjoyable working environment. It is indeed with a great sense of pleasure and immense sense of gratitude that we acknowledge the help of these individuals.

Ms. Neetu , Ms. Ritika Rathi

CERTIFICATE

This is to certify that the project-2 report entitled “ OBJECT RECOGNITION AND CLASSIFICATION” submitted by Ms. Ritika Rathi and Ms. Neetu, as a partial fulfillment for the requirement of B. Tech. VI Semester examination of the School of Engineering and Technology, Mody University of Science and Technology, Lakshmangarh for the academic session 2022-2023 is an original project work carried out under the supervision and guidance of Dr. Sanjeev Patwa has undergone the requisite duration as prescribed by the institution for the project work.

PROJECT GUIDE:

Signature:

Name:

Date:

HEAD OF DEPARTMENT

Signature:

Name:

Date:

EXAMINER-I:

Signature:

Name:

Date:

EXAMINER-II

Signature:

Name:

Date:

ABSTRACT

Real-time item detection and identification are the focus of the project. Using the Open CV package and pre-trained Tensor flow, it will be developed in Python. SSD is the algorithm in use.

A single convolutional neural network is used in the SSD, sometimes referred to as Single Shot Detector. It functions in conjunction with the bounding boxes and extracted characteristics. For constructing default bounding boxes around the observed objects, SSD supports more aspect ratios. SSD enclosures can more precisely and tightly fit around the objects.

So, we are using a pre-trained deep learning architecture based on Tensor Flow for this project.

We must use J processes for picture segmentation, image detection, and identification in order to complete the project.

We categorise or simply identify the object in the image when performing image identification. which we are doing using the Picture Net dataset and the Mobile Net deep learning method.

In object detection, we merely identify the region of the image where an object is visible. We use Mobile Net and SSD for detection.

It is the separation of the foreground and background in an image.

We used the Anaconda Package and Jupiter Notebook as our development environments for this solution. We also needed the Open CV library, which is installed using the pip command, and the matplotlib library for static, animated, and interactive Python visualisations.

The project will then be further coded utilising SSD Mobile Net VJ and Tensor Flow Frozen Model, and at its conclusion, we will be able to recognise and categorise static images, mp4 videos, as well as through webcams.

We will demonstrate the study and use of Real-Time Object detection utilising SSD, one of the quickest object detection algorithms, with the aid of this project. The project's objective is to evaluate various models' expertise in this field.

Table of Contents

S .No.	Topics	Page no.
1.	Introduction	2
1.1	Existing system	3
1.2	Proposed System	3
1.3	Problem statement	3
1.4	Application	4
1.5	Purpose	4
2.	Survey of technology	5-12
3.	Requirement and Analysis	13
3.1	Requirement specification	13
3.2	System Requirement	14
3.3	Planning and scheduling	15
5.	System Design	16-21
6.	Source Code	22
7.	Result (screenshot)	26
8.	Individual Contribution	29
9.	Conclusion	30
9.1	Future scope	30

Chapter1: INTRODUCTION

1. INTRODUCTION

The amount of image data in the world is currently increasing dramatically. Yet, the vast bulk of these images are stored in internet clouds. To effectively manage such enormous amounts of data, we need some understanding of the data's contents. Automated image processing is useful for many tasks that need image recognition or captioning. Images files may include items that can be found and identified automatically.

Modern object detectors use feature extractor and feature classifier, just like traditional object detectors. These two elements can be understood independently, contrary to how they are viewed in contemporary object detectors.

These systems locate each object independently by creating a bounding box around it in addition to identifying and classifying every object in an image. Hence, real-time object detection is more challenging than image recognition using traditional computer vision detectors. The SSD, also known as Single Shot Detectors, is the fastest of the other types since it accurately detects the object. Its speed makes it suitable for applications that require speedy detection, such people counting, autonomous vehicles, and other similar activities, even though it is not the most accurate model when compared to the others.

The goal of object detection is to identify every instance of a known class of items in a picture, such as people, cars, or faces. Even though there are often few instances of the object in the image, there are a vast array of locations and scales where they might appear that must be investigated. Each image detection is supplied together with some kind of pose data. This is as easy to understand as the object's location, scale, or the bounding box-defined extent of the object. In some other circumstances, the pose information is more specific and includes the linear or non-linear transformation's parameters. For instance, in addition to computing the bounding box of the face, a face detector may compute the locations of the eyes, nose, and mouth. Figure 1 displays an illustration of a bicycle detection in an image that pinpoints the locations of specific components. A three-dimensional transformation that specifies the object's position in relation to the camera can also be used to describe the pose. A model for a class of objects is always built by object detection systems from a set of training instances. One example may be sufficient in the case of a fixed hard object in an image, but more typically, numerous training instances are required to adequately capture some features of class variability.

1.1 EXISTING SYSTEM

Work on object detection using conventional computer vision methods has been extensive (sliding windows, deformable part models). They are not as accurate as deep learning-based methods, though. Two main classes of methods—two stage detection (RCNN [1], Fast [2], and Faster [3]) and unified detection—are the most common deep learning-based techniques (Yolo [4], SSD [5]). This is an explanation of the key ideas behind these strategies.

1.2 PROPOSED SYSTEM

Computer vision, the study of computer and software systems that can detect and comprehend images and scenes, is one of the key areas of artificial intelligence. Other components of computer vision include picture identification, object detection, image synthesis, image super-resolution, and other features. Because of the numerous real-world applications, object detection is perhaps the element of computer vision that is most frequently encountered. The ability of software systems to find and identify individual items inside an image or scene is referred to as object detection. It is frequently used in security systems, autonomous automobiles, web photos, pedestrian counting, face detection, vehicle detection, and pedestrian counting.

There are several applications for object detection in a wide range of professional domains. Object detection will undoubtedly be used in a broad range of inventive and remarkable ways, just like every other computer technology, thanks to the efforts of software developers and computer programmers.

Using contemporary object identification techniques in apps and systems and creating new applications based on them are not simple tasks. Traditional algorithms, such as those supported by OpenCV, a well-known computer vision library, were used in the early implementation of object detection. These traditional algorithms, however, lacked the performance necessary to operate effectively in a variety of situations.

1.3 Problem Statement

Before a decade, there were numerous issues with computer vision that were saturating its accuracy. Yet, the accuracy of these issues significantly increased with the development of deep learning algorithms. One of the biggest issues was picture classification, which is

disputed as a method of determining the class of the image. Image localization is a slightly challenging task where the system must estimate the class of the object's placement in the image given a single object in the image (a bounding box around the object). Classification and detection are both involved in the more challenging problem of object detection (the subject of this project). In this instance, an image will be the input to the system, and the system will produce a bounding box that corresponds to every object in the image, along with the class of each object in the box.

1.4 Applications

Face detection is a popular use of object detection that is present in practically all smartphone cameras. Where a variety of objects need to be detected for autonomous driving, a more generalised (multi-class) application can be utilised. Also, it is crucial in surveillance systems. These systems can be used for other tasks like pose estimation, where the first stage of the pipeline is to detect the object and the second is to estimate pose in the discovered region. It may be used to track objects, making robots and medical applications possible.

A wide range of applications, including video surveillance, activity recognition, road condition monitoring, airport safety, monitoring of protection along sea borders, etc., have made use of moving object detection. This technology has the ability to recognise and track things in pictures and videos. Object detection, commonly referred to as object recognition, has several uses including security systems, self-driving cars, pedestrian counting, vehicle recognition, and a lot more.

1.5 PURPOSE

Finding one or more useful targets from still images or video data is the primary goal of object detection. A wide range of significant techniques, including image processing, pattern recognition, artificial intelligence, and machine learning, are all included in it completely. The fundamental objective of object detection is to locate instances of each object in digital images or real-world situations, separate them, and analyse their necessary properties for in-the-moment predictions.

Chapter2 : SURVEY OF TECHNOLOGIES

What is AI?

Artificial intelligence, which includes replicating cognitive processes like perception, learning, and problem-solving, is a broad term for systems and algorithms that can emulate human intelligence. Deep learning (DL) and machine learning are branches of AI.

Modern web search engines, voice-activated personal assistants, self-driving cars, and recommendation systems like those used by Spotify and Netflix are some examples of practical applications of AI.

We have achieved two of the four stages or types of AI; the other two are still in the theoretical stage.

4 types of AI

The four forms of AI are reactive machines, limited memory, theory of mind, and self-awareness, going from most basic to most complex.

Reactive machines are able to carry out fundamental operations in response to an input. At this stage of AI, there is no such thing as "learning"—the system is trained to perform a specific task or set of tasks and never strays from that. These are entirely reactive machines that are incapable of storing inputs, operating outside of a certain environment, or evolving over time.

Most recommendation engines, IBM's Deep Blue chess AI, and Google's AlphaGo AI are examples of reactive machines (arguably the best Go player in the world).

AI systems with a small amount of memory can store information about its actions and judgements, as well as information that is sent to it, and analyse that data later to get better over time. As learning requires a small amount of memory, here is where "machine learning" actually starts.

These are the most advanced AIs we have created to date since limited memory AIs can get better over time. Self-driving cars, virtual voice assistants, and chatbots are a few examples.

The first of the two more sophisticated and (as of this writing) theoretical sorts of AI that we haven't yet developed is theory of mind. At this level, AIs would start to comprehend human ideas and feelings and begin meaningful interactions with us. Instead of the current one-way interactions humans have with a variety of less developed AIs, this relationship between humans and AI is reciprocal.

The term "theory of mind," which is taken from psychology, refers to an AI's comprehension that people have ideas and emotions, which in turn influence the AI's behavior.

For many AI engineers, self-awareness—the state in which AIs have human-level consciousness and are conscious of their existence as sentient entities with similar goals and motivations to humans—is the ultimate aim. Self-aware AIs are currently only a concept seen in science fiction.

What is ML?

Machine learning is a branch of artificial intelligence (AI) that belongs to the "limited memory" category and allows the AI (machine) to learn and advance over time.

The three main categories of machine learning algorithms include reinforcement learning, unsupervised learning, and a number of other methods.

Three distinct kinds of machine learning algorithms similar to the many AI subtypes, there are various machine learning subtypes that range in complexity. Although there are other kinds of machine learning algorithms as well, these three are the most common and are often combined or based on them.

The simplest of them is **supervised learning**, which is when an AI is actively supervised during the learning process, as it states on the box. The computer will be given a large amount of data to process and learn from by researchers or data scientists, as well as some sample findings that the data should yield (more formally referred to as inputs and desired outputs).

An agent that can forecast outcomes based on fresh incoming data is the end result of supervised learning. By saving and repeatedly reexamining these predictions, the machine may continue to hone its learning, thereby increasing its accuracy.

Applications for supervised machine learning include spam detection, media recommendation systems, image identification, and predictive analytics.

There is no human assistance provided during the learning process in **unsupervised learning**. When given a large amount of data to study, the agent finds patterns on its own. Because robots can identify more and different patterns in any given piece of data than humans, this kind of analysis can be quite beneficial. Unsupervised machine learning (ML) can develop over time, much like supervised machine learning.

Applications for unsupervised machine learning include things like identifying client categories in marketing data, imaging in the medical field, and anomaly detection.

Given that no data set is provided to train the system, **reinforcement learning** is the most complicated of these three methods. Instead, the agent gains knowledge through interaction with the environment it is put in. It develops over time by honing its reactions to maximise favourable rewards since it obtains positive or negative incentives depending on the activities it does.

Self-improving industrial robots, automated stock trading, cutting-edge recommendation engines, and bid optimisation for optimising ad expenditure are a few examples of applications for reinforcement learning.

What is DL?

In order to replicate human brain networks, deep learning (DL), a subtype of machine learning, does not require pre-processed input. Huge amounts of unstructured data can be ingested, processed, and analysed by deep learning algorithms so they can learn without any human involvement.

An algorithm for deep learning can get better over time, much like other kinds of machine learning.

Deep learning is now being used to improve computer vision, facial recognition, and natural language processing, among other useful applications.

Object detection is used in many fields to reliably identify various types of things. Many studies have been conducted on some of the most well-known object identification algorithms, including RCNN, ResNet, YOLO, and R-FCN.

OpenCV

OpenCV is a sizable open-source library for image processing, machine learning, and computer vision. It now plays a significant part in real-time operation, which is crucial in modern systems. With it, one may analyse pictures and movies to find faces, objects, and even human handwriting. Python is able to handle the OpenCV array structure for analysis when it is integrated with different libraries, such as NumPy. We use vector space and apply mathematical operations on these features to identify the visual pattern and its numerous features.

OpenCV's initial release was 1.0. OpenCV is free for both academic and commercial use because it is distributed under a BSD licence. It is compatible with Windows, Linux, Mac OS, iOS, and Android and offers C++, C, Python, and Java interfaces. Real-time applications for improved processing efficiency were the primary consideration when OpenCV was developed. Everything is written in C/C++ that has been optimised to take advantage of multi-core processing.

Applications of OpenCV: There are lots of applications which are solved using OpenCV, some of them are listed below

- face recognition
- Automated inspection and surveillance
- number of people – count (foot traffic in a mall, etc)
- Vehicle counting on highways along with their speeds
- Interactive art installations
- Anomaly (defect) detection in the manufacturing process (the odd defective products)
- Street view image stitching
- Video/image search and retrieval
- Robot and driver-less car navigation and control
- object recognition
- Medical image analysis
- Movies – 3D structure from motion
- TV Channels advertisement recognition

OpenCV Functionality

- Image/video I/O, processing, display (core, imgproc, highGui)
- Object/feature detection (objdetect, features2d, nonfree)
- Geometry-based monocular or stereo computer vision (calib3d, stitching, videostab)
- Computational photography (photo, video, superres)
- Machine learning & clustering (ml, flann)
- CUDA acceleration (gpu)

Image-Processing

Image processing is a technique for applying various operations to a picture in order to improve it or to draw out some relevant information from it.

According to the simplest definition of image processing, it is the study and alteration of a digital image, particularly to enhance its quality.

Digital-Image

An image can be thought of as a two-dimensional function, or $f(x, y)$, where x and y are spatial (plane) coordinates. The intensity or grey level of the picture at any given position is determined by the amplitude of the function at any given pair of coordinates (x, y) . In other words, an image is nothing more than a two-dimensional matrix (or three-dimensional in the case of coloured images) that is defined by the mathematical function $f(x, y)$. At any position in an image, the pixel value gives the brightness and colour of that particular pixel.

In essence, image processing is signal processing where the input is an image and the output is an image or a set of characteristics that meet the requirements for that picture.

Basically, image processing involves the following three steps:

1. bringing in the picture
2. Examining and modifying the picture
3. A report or image analysis-based output whose results can be changed.

CNN

Artificial neural networks do incredibly well in machine learning. Artificial neural networks are used to classify words, audio, and images among other things. Various forms of neural networks are employed for various tasks. For example, to predict the order of words, recurrent neural networks—more specifically, an LSTM—are used. Similarly, to classify images, convolution neural networks are employed. We're going to create the fundamental building element for CNN in this blog.

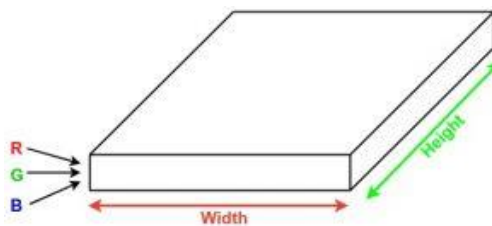
Let's first review the basic concepts of neural networks before delving into the Convolution Neural Network. There are three different sorts of layers in a typical neural network:

1. **Input Layers:** That is the layer where we input data into our model. The entire number of characteristics in our data is equal to the number of neurons in this layer (number of pixels in the case of an image).

2. **Hidden Layer:** The hidden layer is then fed the input from the input layer. Depending on our model and the volume of the data, there may be numerous hidden levels. The number of neurons in each hidden layer might vary, but they are typically more than the number of features. Each layer's output is calculated by multiplying the output of the layer below it by its learnable weights, adding learnable biases, and then computing the activation function, which makes the network nonlinear.
3. **Output Layer:** The output of each class is then converted into the probability score for each class using a logistic function, such as sigmoid or SoftMax, using the data from the hidden layer as input.

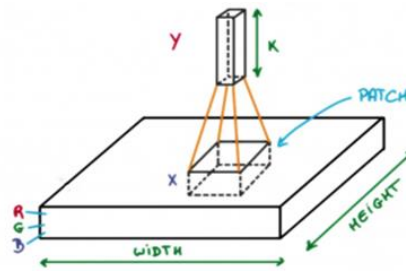
Neural networks that share their parameters are known as convolution neural networks or convnets. Think of having an image. It can be visualised as a cuboid with its length, breadth (the image's dimension), and height (as images generally have red, green, and blue channels).

CNN LAYER



Consider running a small neural network with, say, k outputs on a small portion of this image and representing the results vertically. Move the neural network across the entire image, and a new image with altered width, height, and depth will appear. We now have extra channels, however they are narrower and taller than the original R, G, and B channels. Convolution is the name of this operation. It will be a normal neural network if the patch size is the same as the image size. This little region means that we have fewer weights.

CNN ARCHITECTURE



Let's now discuss some of the math that goes into the convolution process as a whole.

- A group of filters that can be learned make up convolution layers (a patch in the above image). Each filter is narrow, tall, and has a depth equal to the input volume (3 if the input layer is image input).
- For instance, suppose we need to perform convolution on a $34 \times 34 \times 3$ -pixel image. The size of filters may be $a \times a \times 3$, where 'a' may be 3, 5, 7, or another tiny number compared to the size of the image.
- In the forward pass, each filter is incrementally moved across the whole input volume using a step known as stride (which can have a value of 2, 3, or even 4 for high-dimensional pictures), and the dot product between the weights of the filters and the input volume patch is computed.
- We will receive a 2-D output for each filter as we slide it. When we stack the filters, we will obtain an output volume with a depth equal to the number of filters. All the filters will be learned by the network.

RCNN

A technique called region suggestions developed by Ross Girshick and colleagues only extracts 2000 regions from the image.

The algorithm recommended using just 2000 areas to classify a substantial number of regions rather than attempting to categorise a huge number of locations. A convolutional neural network receives these 2000 region suggestions and square-shaped-bent/twisted twists them before feeding them in. The result is a 4096-dimensional feature vector. The output dense layer is composed of the numerous features that were extracted from the input image by the CNN, which serves as a feature extractor. To determine whether the object is present within the suggested candidate region, the obtained features are subsequently fed into an SVM.

The problems with R-CNN are that it cannot be applied in real-world scenarios and that it still takes a long time to train the network because it must classify 2000 region proposals every image.

There is therefore no learning happening at that time. This might lead to the creation of ideas for underdeveloped regions.

YOLO (You Only Look Once)

The most recent version of YOLO, also known as You Only Look Once, is Yolo-v3, which is referred to as DarkNet-53 in custom CNN architecture. The original and earliest Yolo v1 design was influenced by Google Net, which created final predictions from a tensor and did down sampling on the observed images. This tensor is created using the Faster R-CNN network's Region of Interest pooling layer. Yolo v2's 30-layer design included 11 new layers that were heavily utilised for object detection together with 19 layers from Darknet-19. This new design offered a quicker and more accurate object detection mechanism, but it had trouble correctly recognising small items that are in the zone of interest.

In order for YOLO to function, a picture must first be split into a SxS grid and m bounding boxes must be assigned to each grid. The network outputs a "a" class probability and offset values for each bounding box that is created. The object in the image is further located using the bounding box that has the class probability over a threshold value. YOLO outperforms other available object detection algorithms by an order of magnitude (45 frames per second).

The YOLO algorithm's restriction and drawback is that it has trouble identifying small things in the image. For instance, it might have trouble identifying a bird in the image. The YOLO algorithm's spatial limitations are to blame for this.

ResNet

ResNet trains hundreds or thousands of layers to produce exceptional performance. By utilising its potent representational ability, many Computer Vision applications, such as face recognition and object detection, have had their performance negatively impacted. Deep neural networks are difficult to train because of the vanishing gradient problem, which happens when a gradient is multiplied repeatedly and shrinks to an infinitesimally small value.

As a result, as the network goes deeper, performance begins to rapidly decline or become saturated. A "identity shortcut connection," which designates the omission of one or more layers, is the main idea behind ResNet.

These feature layers' sizes gradually shrunk, allowing prediction of the detection on different scales. Experimental results show that the SSD's underlying convolution network is replaced by a residual network when the input size is 300 or 320, which lowers rather than raises accuracy.

R-FCN

While RFCN also requires region suggestions, it removes the fully convolutional layers after ROI pooling.

RPN is slowed by the FC layers' laborious and time-consuming procedure after pooling, which does not share the ROI. Connections are multiplied by the FC levels, increasing complexity. The same set of score maps will be used to calculate the average vote for each region proposal, which is an easy calculation. These score maps are constructed from convolutional feature maps that have been trained to recognise particular characteristics of each object. Using competitive MAP, RFC performs faster than Faster RCNN.

Fast RCNN and Faster RCNN

More advancements in the area of object detection have been made by Fast RCNN and Faster R. They use convolutional layers with initialised pretraining for ImageNet classification to retrieve region-independent data.

For classification, multi-layer perceptrons (MLP) and features are used.

Since Fast RCNN extracts features from the image before processing regions, it is faster than RCNN at detecting objects. It replaces the SVM with a SoftMax layer, which facilitates in the extension of a neural network, rather than creating a new model. The Faster RCNN Selective Search approach has been replaced by the Region Proposal Network (RPN), which uses feature maps to try and learn an object's proposal. RPN uses the CNN-produced feature maps to recommend the regions after receiving them. These region proposals must be produced using K numbers of anchor boxes for each location on the feature maps.

Chapter 3: REQUIREMENTS AND ANALYSIS

3.1 REQUIREMENT SPECIFICATION

Install Python on your computer system

1. Install Anaconda and anaconda prompt.
2. Python code implementation on Jupyter Notebook.

3.3 SYSTEM REQUIREMENT:

Install Python on your computer system

1. Install ImageAI and its dependencies like tensorflow, Numpy, OpenCV, etc.
2. Download the Object Detection model file(Retinanet)

Steps to be followed :-

- 1) Download and install anaconda
- 2) Install the following dependencies via pip:

i. Tensorflow:

An open-source software package called Tensorflow is used for dataflow and differentiable programming on a variety of tasks. It is a library for symbolic maths and is also used in machine learning programmes like neural networks, etc. Google uses it for both research and production.

The Google Brain team creates Tensorflow for use at Google. On November 9, 2015, it is made available under the Apache License 2.0.

The second-generation Google Brain system is called Tensorflow.

Tensorflow's initial version was released on February 11, 2017.

Tensorflow can run on several CPUs and GPUs while the standard implementation only supports a single device (with optional CUDA and SYCL extensions for general-purpose computing on graphics processing units). TensorFlow is accessible on a number of operating systems, including 64-bit Linux, macOS, Windows, and mobile operating systems like Android and iOS.

Tensorflow's architecture makes it simple to deploy compute across a range of platforms (CPUs, GPUs, and TPUs), from PCs to server clusters to mobile and edge devices.

Stateful dataflow graphs are used to represent calculations in Tensorflow. The operations that these neural networks carry out on multidimensional data arrays, known as tensors, are where the name Tensorflow originates.

pip install tensorflow -command

ii. Numpy:

The Python programming language's NumPy module adds support for massive, multidimensional arrays and matrices as well as a sizable number of high-level mathematical operations that may be performed on these arrays. Jim Hugunin originally developed Numeric, the predecessor of NumPy, with assistance from a number of developers. Travis Olphant developed NumPy in 2005 by integrating crunching Numarray features into Numeric and modifying the extension. Several people have contributed to the open-source programme NumPy.

pip install numpy -command

iii. OpenCV:

A set of programming tools called OpenCV is primarily focused on real-time computer vision. It was initially created by Intel and is afterwards sponsored by Willow Garage and Itseez. Cross-platform and open-source BSD licence make the library free to use.

pip install opencv-python -command

iv. Matplotlib:

Python's graphing toolkit, Matplotlib, and its NumPy extension support numerical maths. It offers a general-purpose object-oriented API for using GUI toolkits like Tkinter.

pip install matplotlib – command

v. ImageAI:

Using pre-trained models that were built on the ImageNet-1000 dataset, ImageAI gives API to identify 1000 different objects in a picture. SqueezeNet, ResNet, InceptionV3, and DenseNet are the model implementations that are offered.

pip3 install imageai –upgrade

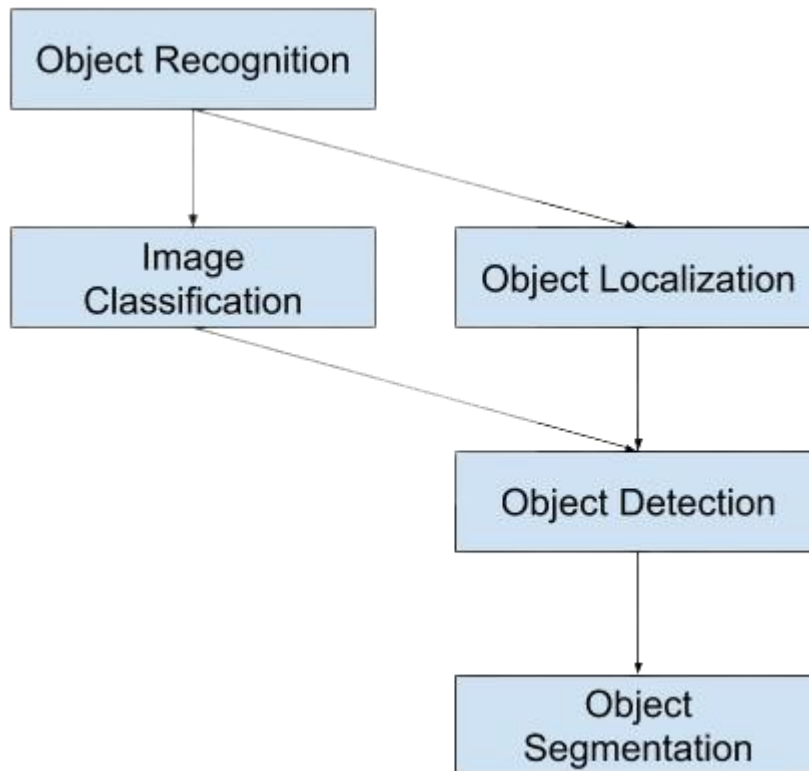
3.3 PLANNING AND SCHEDULING

TASKS	3 february- 18 february	18 February- 16 march	16march- 25 March	25 march- 15 April	15 April- 29 april
Introduction to project and understanding problem statement					
Starting with importing of libraries					
Completing the importing of libraries					
Model training and checking for the result					
Result analysis ana testing of project and report writing					

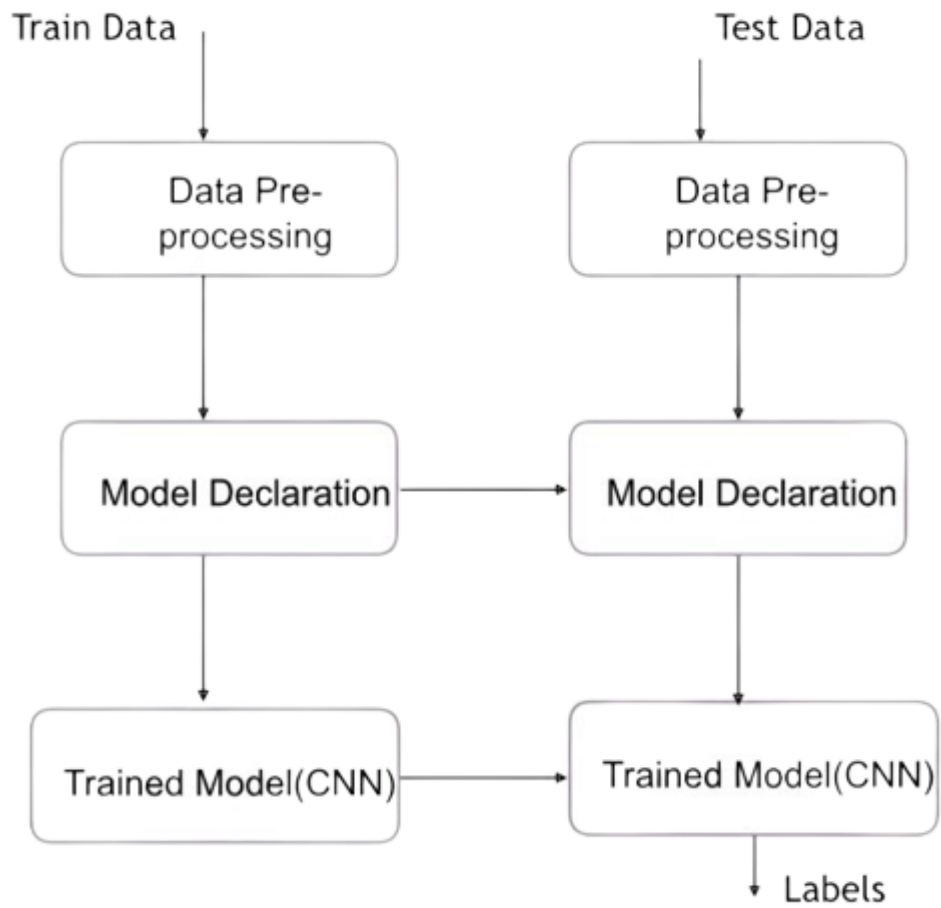
- A Gantt chart is a form of bar chart that shows the timeline for a project.
- The vertical axis of this graph includes the time interval and the tasks that must be completed axis horizontal.
- The length of each activity is represented by the width of the horizontal bar in the graph. The start and finish dates for the project's final and summary items are shown in a Gantt chart.
- Plan for the summary and closing paragraphs. Dependencies are also shown in a modern Gantt chart (that is, a network of priorities). connections between various activities. The Gantt chart allows you to see the status of your current schedule. Utilize the vertical "TODAY" line and % shading as displayed above.
- Bar charts and Gantt charts are occasionally used interchangeably.
- Gantt charts are usually first created with an initial start time. Each task is scheduled to start as soon as the prerequisites are met.
- This method maximizes the flextime available for all tasks.

Chapter 4: SYSTEM DESIGN

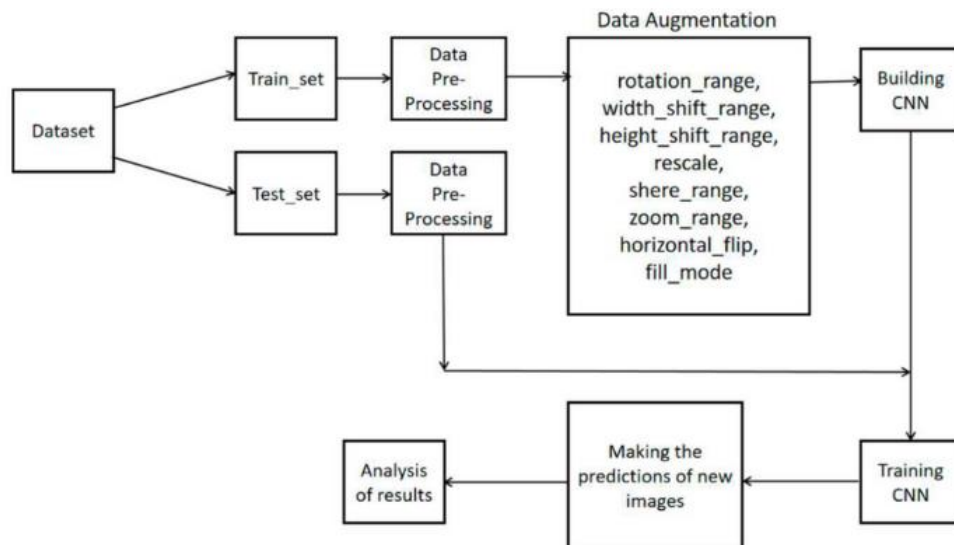
FLOW CHART



BLOCK DIAGRAM



Project Design



Dataset – It contains the Different data set like Images and Videos about the objects which need to detect and helps to train the System to detect the custom object.

Train_set – Train_set are the set of data which is collected for to train the machine to detect the custom object.

Data Pre-processing – Now the Train_set will be in processing to train the system to detect a custom object with the help of data pre-processing.

Building CNN – CNN is a train data sets about the Data pre-processing which will be executed to detect the custom object.

Training CNN – This file contains the processed data set and the Pre-processing data.

Analysis of result – With the help of Building CNN and Training CNN we can detect the custom object for which our system is trained.

Chapter 5: IMPLEMENTATION AND TRAINING

The Google research team developed the source deep learning framework. With MobilNet v1, the quickest model for object detection, we used SSD, also known as Single Shot Multi-Box Detector.

The multi-box detector permits simultaneous detection of many objects in the scene by creating m bounding boxes around each object in the image. We created this model using a low-cost GPU and a network that has already been trained using the COCO dataset. We have expanded our training set and adjusted a few parameters to further improve performance.

Convolutional layers can be replaced by depth-wise separable convolutions in MobileNet V1, despite the fact that they are very expensive to compute but are necessary for computer vision applications. The convolution layer has two distinct purposes. An initial depth-wise convolution layer filters the input.

These values are then combined in a 1×1 convolution to produce the new features. The depth wise and pointwise convolutions combine to create the depth wise separable convolution block.

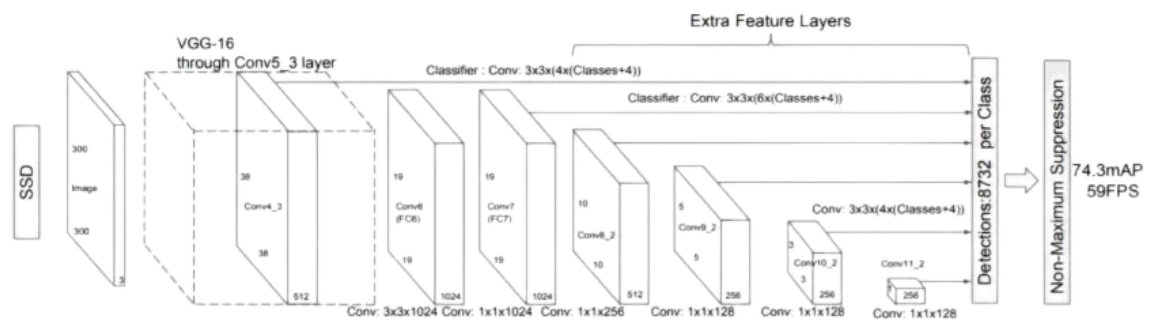
It accomplishes the same task as the convolution method but does it much more quickly.

3.1 SSD

To locate objects in pictures or movies, the Single Shot MultiBox Detector deep learning model is utilised. Single Shot Detector is a simple answer to the problem, but so far it has proven to be fairly effective.

The two components of an SSD are the Backbone Model and the SSD Head. Backbone Model is a pre-trained image categorization network that functions as a feature extractor. Usually, the fully connected categorization layer is removed from the model. This backbone has an additional set of convolutional layers called the SSD Head, whose outputs are interpreted as the bounding boxes and classifications of objects in the spatial positions of the final layer's activations.

SSD ARCHITECTURE



The SSD object detection composes of 2 parts:

1. Extract feature maps, and
2. Apply convolution filters to detect objects.

The image is divided into grids by SSD, and each grid cell is in charge of identifying things in that area of the image. If no object is found, we output "nothing" or, to be more precise, "0," which denotes that no object was located.

SSD only recognises items from one layer. In reality, it employs many layers (multi-scale feature maps) for independently detecting objects. The resolution of the feature maps declines as CNN gradually reduces the spatial dimension. For the purpose of detecting larger-scale objects, SSD uses lower resolution layers. For the bigger size item, for instance, the 44 feature maps are employed.

Deep convolutional neural networks can categorise objects strongly against the realting transformation because to the cascade of pooling operations and non-linear activation. The local search window in sliding window detection on SSD is the receptive field.

Similar to other sliding window approaches, the SSD's search has a finite resolution that is determined by the convolution's stride and the pooling mechanism. As a result of the off-target receptive field, SSD will get erroneous data.

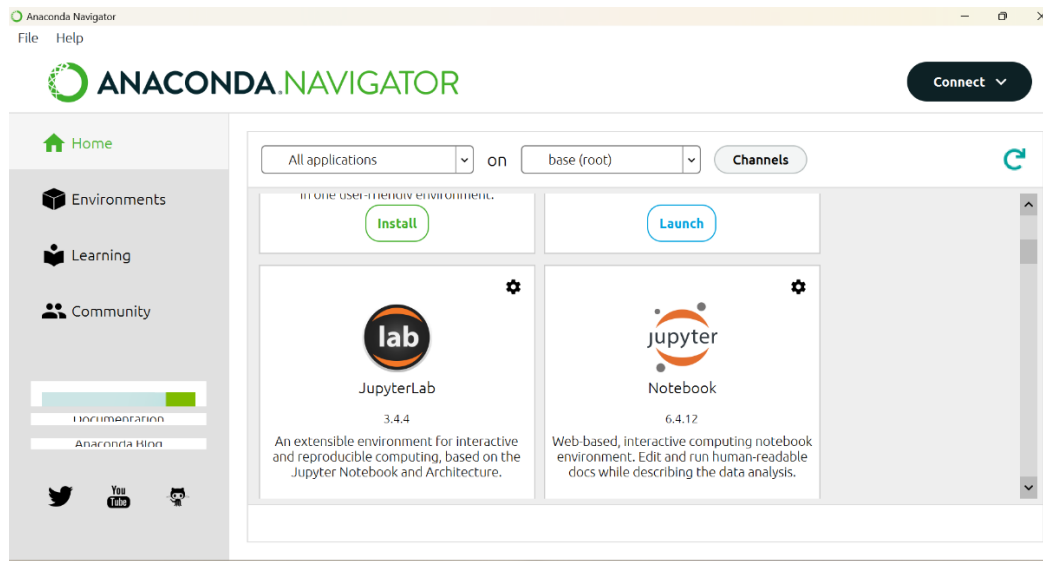
Deep convolutional neural networks are capable of predicting an object's class as well as its precise location. A vector of four floating-point values that SSD can map to can indicate the bounding box. When the unwanted forms are gone, the detection is far more accurate and uses a lot less computing resources.

Because to SSD, the classification and localization tasks can share features. The very last layer of these two projects is the only distinction between them. This considerably lowers the cost of computation while also assisting the network in learning the properties.

ADVANTAGES OF SSD

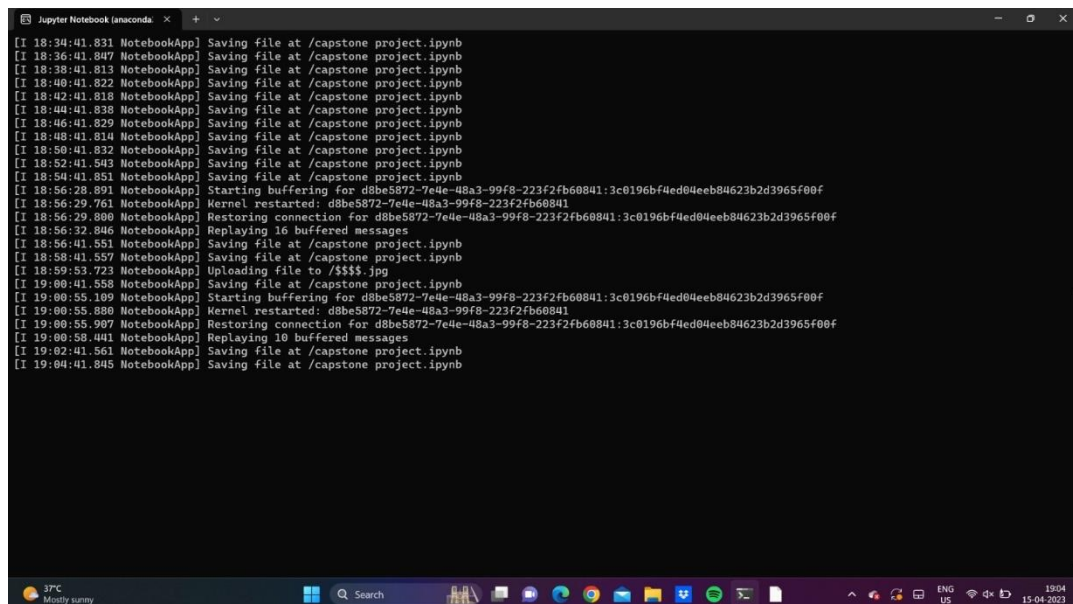
- SSD performs the same function as two-shot detectors but costs less overall. In circumstances where there are few resources available, they perform far better.
- In comparison to other solutions, it has a very tiny drop in accuracy. In comparison to SSD500, which achieved 22 FPS with mAP 76.9%, SSD300 recorded 59 FPS with mAP 74.3%.
- SSD can easily outperform Faster R-CNN and R-FCN in accuracy for large objects with lighter and shorter extractors.
- The Single Shot Detector network combines predictions from several feature maps with various resolutions to naturally handle objects of various sizes.

Anaconda navigator

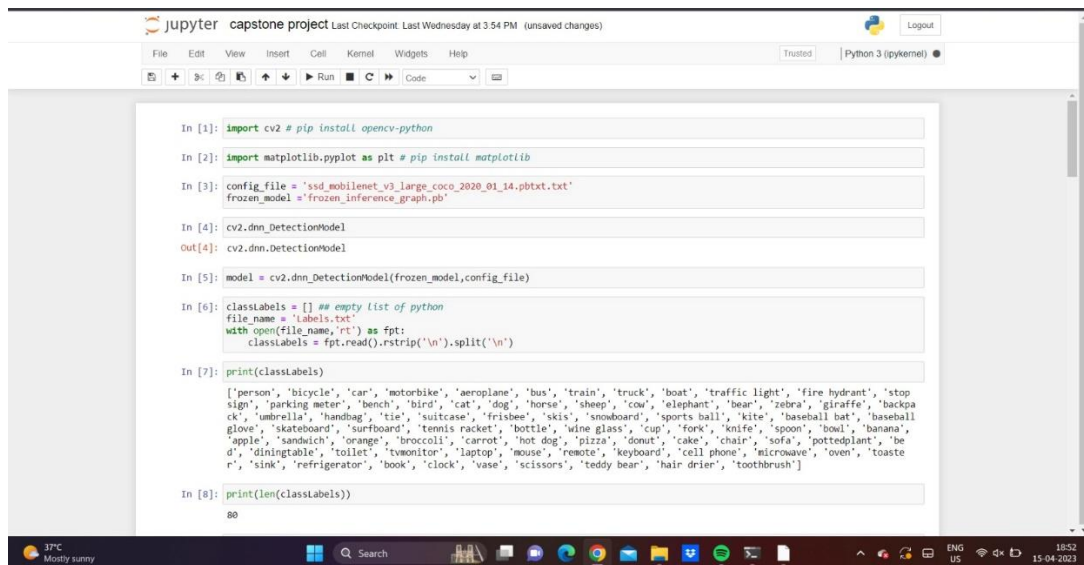


For implementation used Anaconda Package and Jupiter Notebook as IDE and we require Open Library which is installed through pip command and matplotlib, pyplot library for static, animated and interactive visualizations in python.

Jupyter Notebook



Import OpenCV



```
In [1]: import cv2 # pip install opencv-python

In [2]: import matplotlib.pyplot as plt # pip install matplotlib

In [3]: config_file = 'ssd_mobilenet_v3_large_coco_2020_01_14.pbtxt.txt'
frozen_model = 'frozen_inference_graph.pb'

In [4]: cv2.dnn_DetectionModel
Out[4]: cv2.dnn.DetectionModel

In [5]: model = cv2.dnn_DetectionModel(frozen_model, config_file)

In [6]: classLabels = [] ## empty list of python
file_name = 'Labels.txt'
with open(file_name, 'rt') as fpt:
    classLabels = fpt.read().rstrip('\n').split('\n')

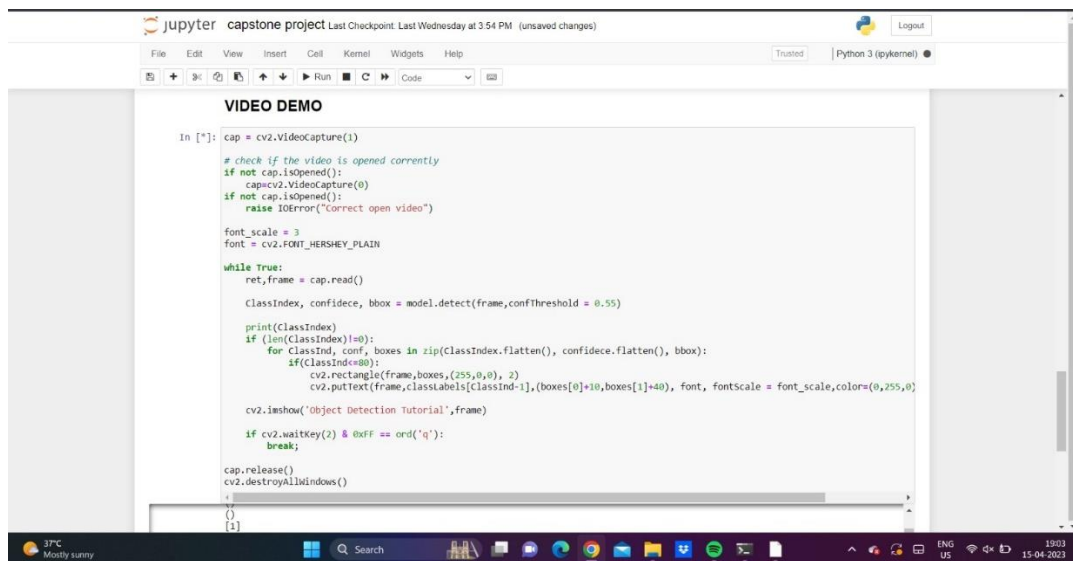
In [7]: print(classLabels)

['person', 'bicycle', 'car', 'motorbike', 'aeroplane', 'bus', 'train', 'truck', 'boat', 'traffic light', 'fire hydrant', 'stop
sign', 'parking meter', 'bench', 'bird', 'cat', 'dog', 'horse', 'sheep', 'cow', 'elephant', 'bear', 'zebra', 'giraffe', 'backpa
ck', 'umbrella', 'handbag', 'tie', 'suitcase', 'frisbee', 'skis', 'snowboard', 'sports ball', 'kite', 'baseball bat', 'baseball
glove', 'skateboard', 'surfboard', 'tennis racket', 'bottle', 'wine glass', 'cup', 'fork', 'knife', 'spoon', 'bowl', 'banana',
apple', 'sandwich', 'orange', 'broccoli', 'carrot', 'hot dog', 'pizza', 'donut', 'cake', 'chair', 'sofa', 'pottedplant', 'be
d', 'diningtable', 'toilet', 'tvmonitor', 'laptop', 'mouse', 'remote', 'keyboard', 'cell phone', 'microwave', 'oven', 'toaste
r', 'sink', 'refrigerator', 'book', 'clock', 'vase', 'scissors', 'teddy bear', 'hair drier', 'toothbrush']

In [8]: print(len(classLabels))

80
```

Project code Snippet



```
VIDEO DEMO

In [1]: cap = cv2.VideoCapture(1)

# check if the video is opened currently
if not cap.isOpened():
    cap = cv2.VideoCapture(0)
if not cap.isOpened():
    raise IOError("Correct open video")

font_scale = 3
font = cv2.FONT_HERSHEY_PLAIN

while True:
    ret, frame = cap.read()
    ClassIndex, confidece, bbox = model.detect(frame, confThreshold = 0.55)

    print(ClassIndex)
    if (len(ClassIndex) != 0):
        for ClassInd, conf, boxes in zip(ClassIndex.flatten(), confidece.flatten(), bbox):
            if (ClassInd != 80):
                cv2.rectangle(frame, boxes, (255, 0, 0), 2)
                cv2.putText(frame, classLabels[ClassInd-1], (boxes[0]+10, boxes[1]+40), font, fontScale = font_scale, color=(0, 255, 0))

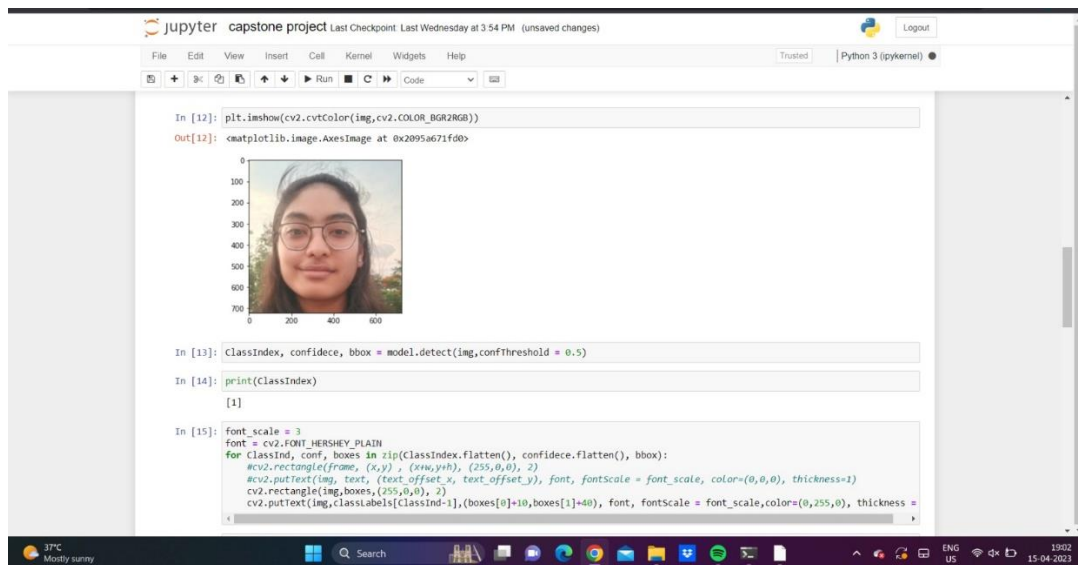
    cv2.imshow('Object Detection Tutorial', frame)

    if cv2.waitKey(2) & 0xFF == ord('q'):
        break

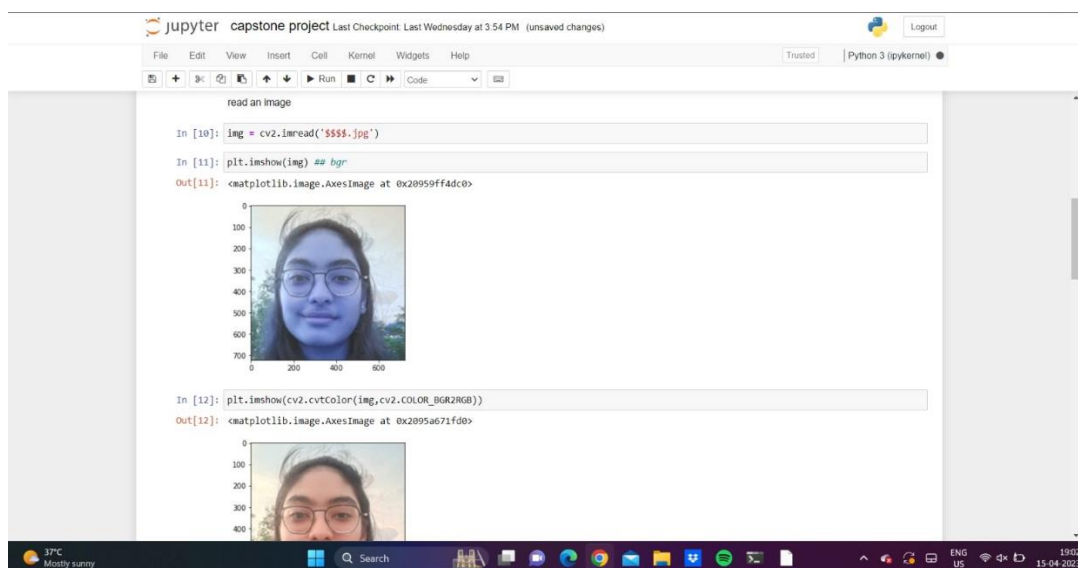
cap.release()
cv2.destroyAllWindows()

[1]
```

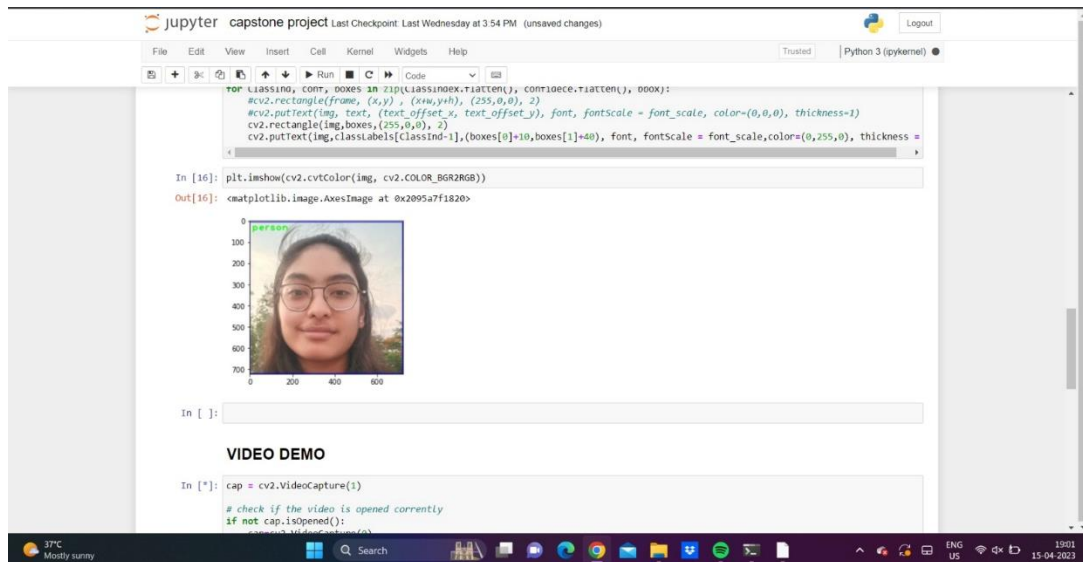
Read Image



Check Background Color



Output



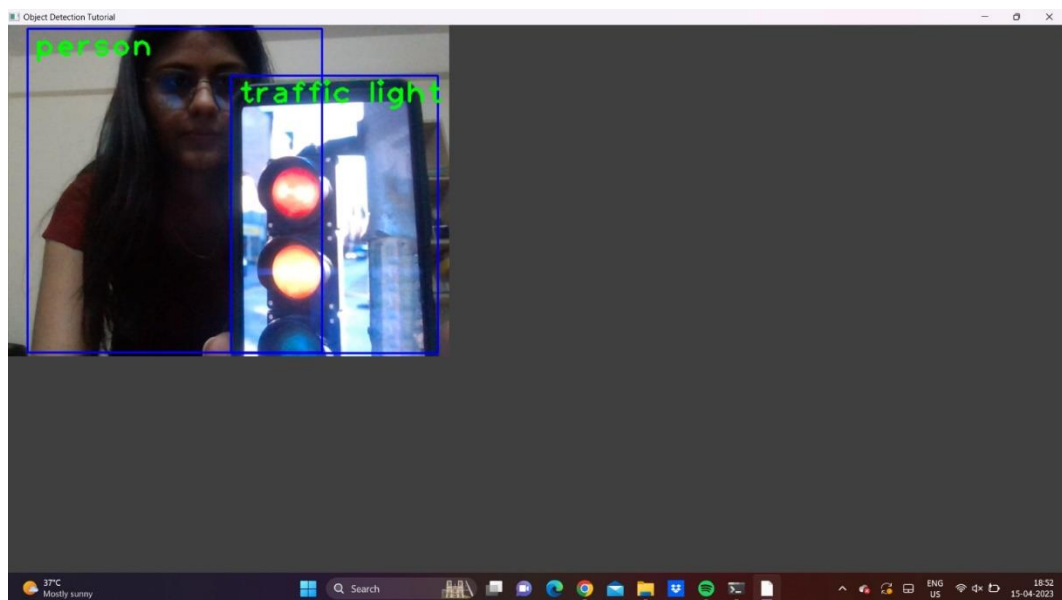
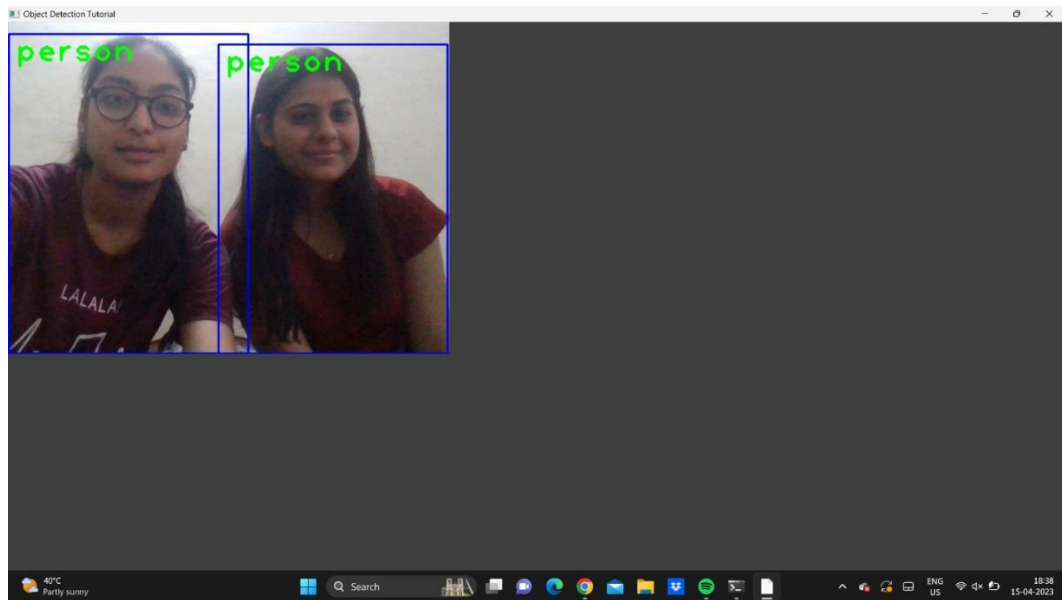
The proposed system accurately detects and recognises the objects in each of the multiple tests with different objects. Python was used to build the project, and it was reviewed multiple times through webcam. It has a decent frame rate. Each frame of the input video is broken up into its component parts, and each frame is then transmitted to our in-house object detector for object detection. The SSD method performs object detection after detection and then receives the bounding box data. Our proposed system was used to analyse the live video, and the output was displayed as bounding boxes along with the class name and confidence level.

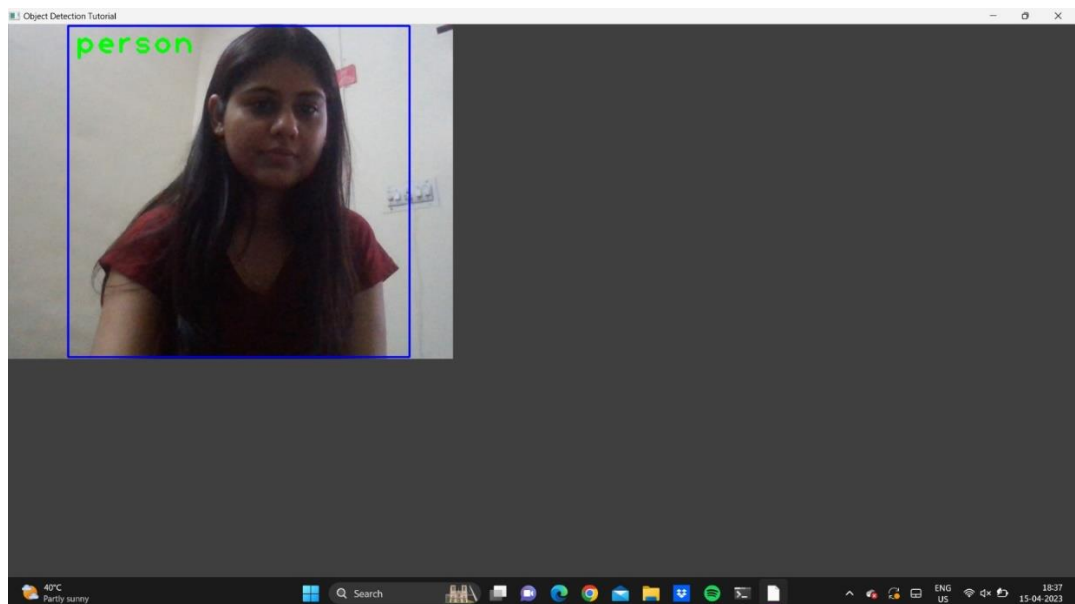
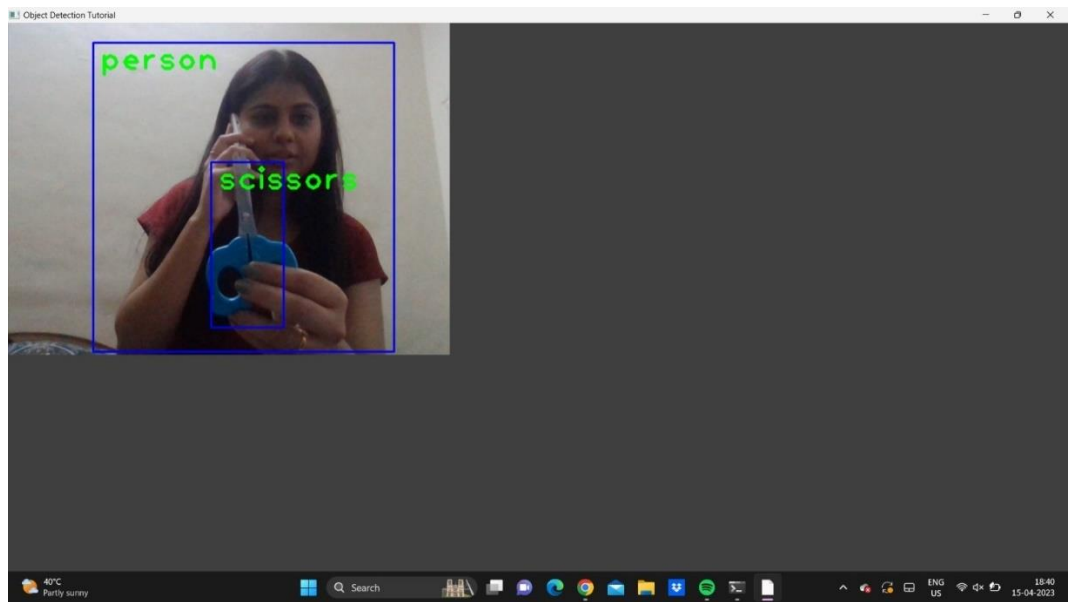
The SSD model with MobileNet v1 has the fastest GPU response time in milliseconds when compared to other object detection models, as shown by the aforementioned graphic. RCNN performed the worst when measured in GPU time. An image resolution of 300 dpi was used for this. Because SSD is the model that can distinguish objects the quickest and in the smallest length of time, it is advantageous in applications like counting people or items. Moreover, SSD is the quickest Object Detection Model in reliably identifying small objects, whereas RCNN and other models like SSD are not.

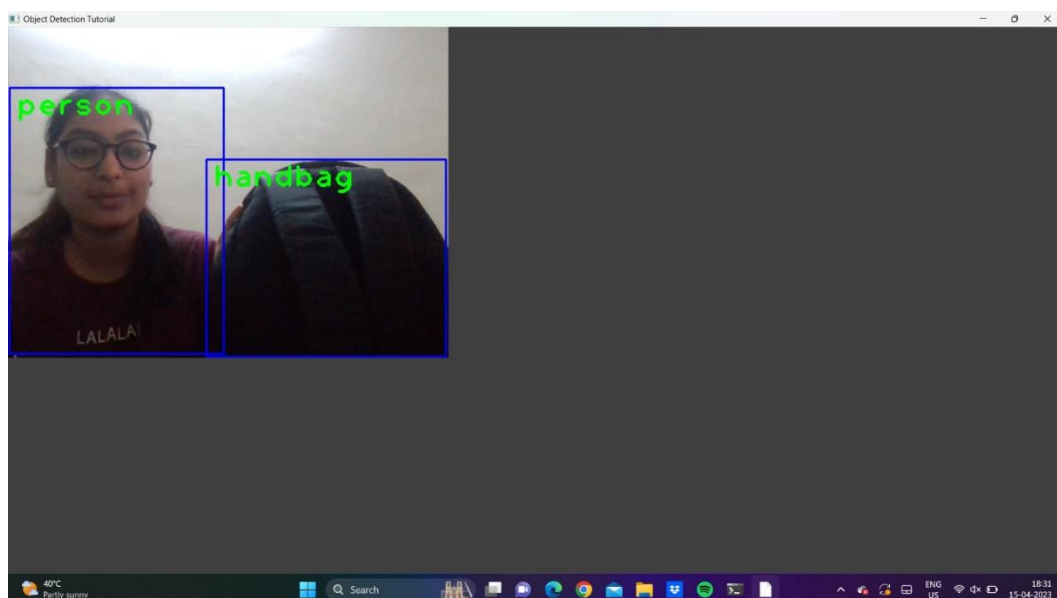
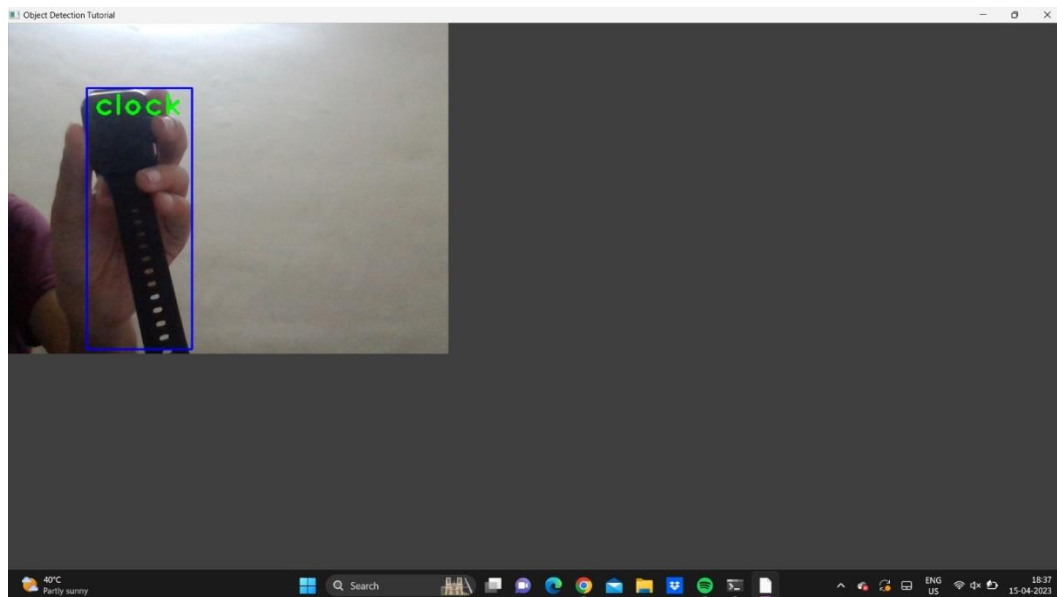
Chapter 7: RESULTS (screenshot)

RESULTS

Result of Implementation







Chapter :8 INDIVIDUAL CONTRIBUTION

We had meetings to decide how to approach the project most effectively. We discussed so much material on this initial level that we came to the conclusion that it was time to organise our thoughts and go to work on the project. We both made our own contributions to the project, and a meeting was later scheduled to compile and go over our data.

Every team member has particular strengths and weaknesses. When they are well-organized and everyone is putting up their best effort, learning teams can be a fruitful way to develop new skills and enhance existing ones.

Ms. Neetu

I contributed to the team project's accomplishment. I provided information and suggestions for the project and encouraged the team's creative efforts. I spend a significant amount of time working with team members on certain project tasks that fit within my area of expertise in order to aid the team in achieving project objectives. As part of the project, each team member was assigned a specific task to fulfil, and eventually all of the contributions were integrated.

Ms. Ritika Rathi

I can help the project by tackling my job in a distinctive and creative way. I believe that my skills and expertise allowed me to solve problems with speed and accuracy. I'm constantly learning new things from the internet and putting them into practise because we are novices. Finally, I contributed by continually setting a good example for our project and making sure that my actions are respectful and insightful.

Chapter 9: CONCLUSION

Using SSD, we can quickly detect objects in this project (single Shot Detector). It helps with the quickest identification of specific visual elements.

Through the use of Single Shot Multi-Box Detector, we are able to identify multiple items concurrently in real time. We have shown that the widely used fully convolutional neural networks may be applied to Real-Time Object Detection systems.

The networks were trained on a low-cost GPU using the MSCOCO dataset in order to increase the accuracy of the objects recognised by increasing the probability of the objects discovered. The model's performance was then assessed. In terms of object localization, SSD fared better than Fast RCNN and Faster RCNN.

Ultimately, we are able to use MobilNet v1 to perform Real-Time Object Detection utilising SSD, which is quicker and more effective than previous object detection techniques.

9.1 FUTURE SCOPE OF THE PROJECT

The project's goal is to create an object recognition system that can identify the 2-D and 3-D objects in a photograph. By improving the global or local properties, Object Detection systems' efficiency can be raised even more. The colour data is ignored in the proposed Object Detection approach, which only uses greyscale images. The colour information from the photographs can be utilised to more accurately identify objects because colour is so essential in robotics. Moreover, a built-in night vision option can be added to tracking devices and CCTV cameras. Due to its extensive coverage and different viewing angles for the tracked objects, multiview tracking can be used to overcome the aforementioned limitations and make the systems completely autonomous.

REFERENCE

1. Agarwal, S., Awan, A., and Roth, D. (2004). Learning to detect objects in images via a sparse, part- based representation. *IEEE Trans. Pattern Anal. Mach. Intel.* 26,1475–1490. doi:10.1109/TPAMI.2004.108
2. Alexe, B., Deselaers, T., and Ferrari, V. (2010). “What is an object?” in *ComputerVision and Pattern Recognition (CVPR)*, 2010 IEEE Conference on (San Francisco, CA: IEEE), 73–80.doi:10.1109/CVPR.2010.5540226
3. Aloimonos, J., Weiss, I., and Bandyopadhyay, A. (1988). Active vision. *Int. J.Comput. Vis.* 1,333–356. doi:10.1007/BF00133571
1. <https://www.geeksforgeeks.org/opencv-overview/>
5. Azzopardi, G., and Petkov, N. (2013). Trainable cosfire filters for keypoint detection and pattern recognition. *IEEE Trans. Pattern Anal. Mach. Intell.* 35, 490–503.doi:10.1109/TPAMI.2012.106
6. Azzopardi, G., and Petkov, N. (2014). Ventral-stream-like shape representation : from pixel intensity values to trainable object-selective cosfire models. *Front.Comput. Neurosci.* 8:80.doi:10.3389/fncom.2014.00080
7. Benbouzid, D., Busa-Fekete, R., and Kegl, B. (2012). “Fast classification using sparse decision dags,” in *Proceedings of the 29th International Conference on MachineLearning (ICML-12)*, ICML ‘12, eds J. Langford and J. Pineau (New York, NY:Omnipress), 951–958.
8. Bengio, Y. (2012). “Deep learning of representations for unsupervised and transfer learning,” in *ICML Unsupervised and Transfer Learning*, Volume 27 of *JMLRProceedings*, eds I. Guyon, G. Dror, V. Lemaire, G. W. Taylor, and D. L. Silver (Bellevue: JMLR.Org), 17–36.
9. Bourdev, L. D., Maji, S., Brox, T., and Malik, J. (2010). “Detecting people using mutually consistent poselet activations,” in *Computer Vision – ECCV2010 – 11th European Conference on Computer Vision*, Heraklion, Crete, Greece, September 5-11, 2010, *Proceedings, Part VI*, Volume 6316 of *Lecture Notes in Computer Science*, eds K. Daniilidis, P. Maragos, and N. Paragios (Heraklion:Springer), 168–181.
10. Bourdev, L. D., and Malik, J. (2009). “Poselets: body part detectors trained using 3dhuman pose annotations,” in *IEEE 12th International Conference on ComputerVision, ICCV 2009*, Kyoto, Japan, September 27 – October 4, 2009 (Kyoto: IEEE), 1365–1372.

11. Cadena, C., Dick, A., and Reid, I. (2015). “A fast, modular scene understanding system using context-aware object detection,” in Robotics and Automation (ICRA), 2015 IEEE International Conference on (Seattle, WA).
12. Correa, M., Hermosilla, G., Verschae, R., and Ruiz-del-Solar, J. (2012). Human detection and identification by robots using thermal and visual information in domestic environments. *J. Intell. Robot Syst.* 66, 223–243. doi:10.1007/s10846-011-9612-2.
13. Dalal, N., and Triggs, B. (2005). “Histograms of oriented gradients for human detection,” in Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on, Vol. 1 (San Diego, CA: IEEE), 886–893. doi:10.1109/CVPR.2005.177.
14. Erhan, D., Szegedy, C., Toshev, A., and Anguelov, D. (2014). “Scalable object detection using deep neural networks,” in Computer Vision and Pattern Recognition Frontiers in Robotics and AI

www.frontiersin.org November 2015.

- <https://docs.python.org/3.8/library/index.html>
- <https://www.geeksforgeeks.org/introduction-to-multi-task-learningmtl-for-deep-learning/>
- <https://www.geeksforgeeks.org/introduction-machine-learning-using-python/>
- <https://www.geeksforgeeks.org/learning-model-building-scikit-learn-python-machine-learning-library/>
- https://github.com/tirthajyoti/Machine-Learning-with-Python/blob/master/Pandas%20and%20Numpy/Numpy_Pandas_Quick.ipynb
- https://github.com/tirthajyoti/Machine-Learning-with-Python/blob/master/OOP_in_ML/Class_MyLinearRegression.ipynb